The Effect of PLD Architecture on Cores

December 5, 1997

Background

Intellectual property (IP) in the form of pre-defined functions, or cores, has been a hot topic in the electronics industry. The ultimate goal is to be able to select cores off-the-shelf in the form of virtual components from multiple sources and combine them into a system-level design using any EDA tool of preference. The resulting design could then be targeted to any silicon vendors process.

That is the ideal situation and someday we may get there. Unfortunately, there are still tool and silicon dependencies today that effect the performance of a core. This is particularly true for programmable logic. A core that has been tuned for a particular programmable logic device (PLD) architecture will achieve higher performance and better utilization than would a generic, synthesized version of the same core.

Architecture, then, plays a critical role in the implementation of cores in programmable logic. This paper will look at how PLD features impact the performance, consistency, power consumption and efficiency of cores. We will also address some aspects of the impact of software.

PLD Architectures

Most PLDs consist of an array of logic cells that contain a look-up-table and a flip-flop. In Xilinx FPGAs for example, the basic element is known as a Configurable Logic Block, or CLB as shown in Figure 1. The logic cells in PLDs are larger than the basic logic element of an ASIC style gate array, and this difference is the main cause for the less efficient and lower performance results obtained for PLDs through a typical synthesis flow.

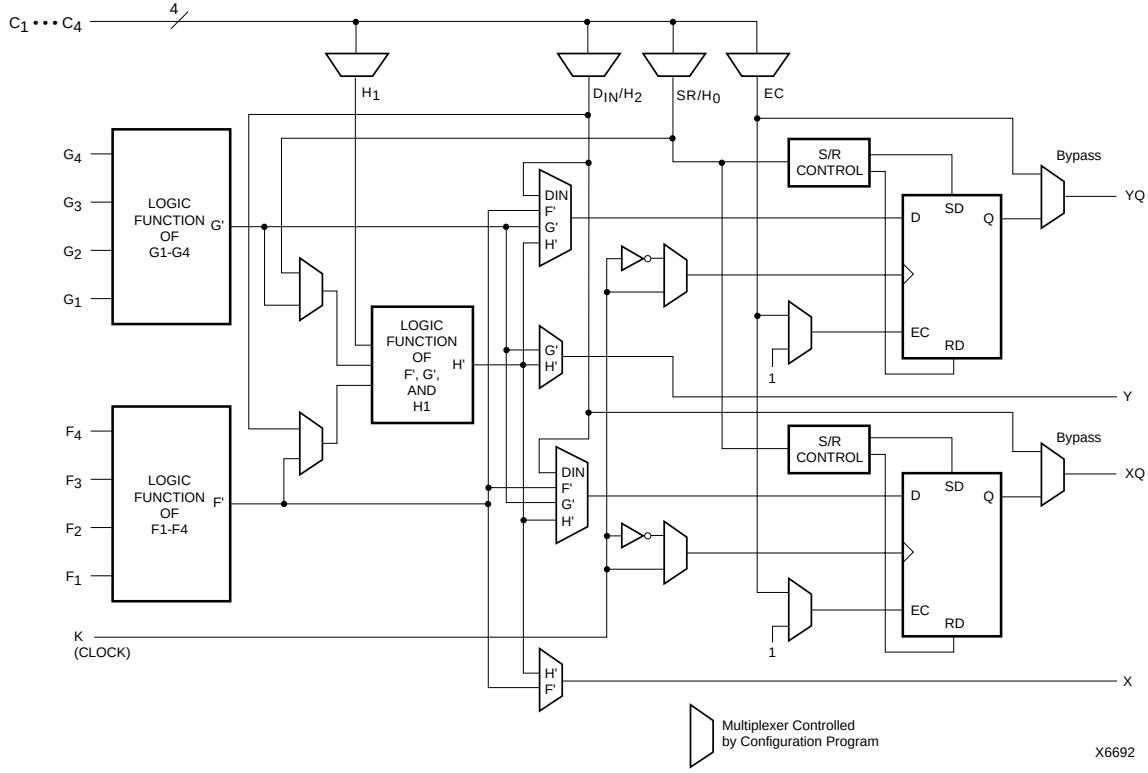

Figure 1: Xilinx XC4000 CLB Structure

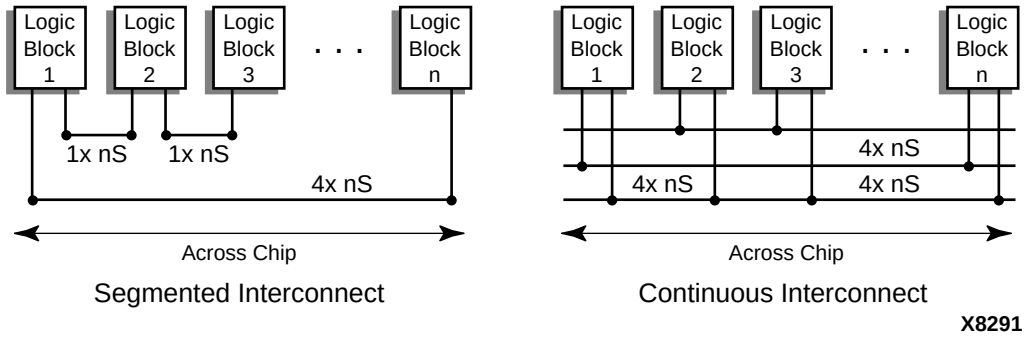

Figure 2: Segmented and Continuous Interconnect

There are differences in each PLD vendor's logic block structure as well as differences in features such as I/O and carry logic, but these involve more detail than will be dealt with here. Instead, we will focus on differences in routing resources and on-chip RAM since these have the most dramatic impact on the use of cores.

Routing Resources

PLDs generally provide routing resources in one of two forms - segmented or continuous (non-segmented) interconnect - as shown in Figure 2.

Xilinx supports segmented interconnect, and also provides global and long-line resources in all of its FPGAs. Segmented interconnect is used for short signals between adjacent CLBs. Long line and global routing resources are used for buses, high fan-out signals, or signals that must travel across the device.

Continuous interconnect architectures rely on long-line resources for signal routing. The touted advantage is to lower the decision-making burden on the place and route tools, reducing overall place and route compilation times. It has, however, a detrimental effect on any design, including those that use cores, as we will see later.

On-Chip RAM

RAM in PLDs appears either distributed across the logic array, or in dedicated, embedded blocks. For example, within each CLB of the Xilinx XC4000 family of FPGAs are two 4-input look-up tables that can be used for logic, or configured as single- or dual-port RAMs in a distributed RAM architecture.

Comparative Analysis

Let's examine, then, the impact that these features have on designs that use cores. We will focus on three important areas: 1) performance consistency, or predictability, 2) efficiency of resource utilization and 3) power consumption.

Performance Consistency

The interconnect scheme of the PLD has considerable impact on how predictable a core's performance will be. Cores designed using relational placement constraints and segmented routing instead of long-line resources can be placed anywhere in the logic array, and will perform exactly as specified regardless of surrounding logic. This is very important when a design uses multiple cores, or cores plus other logic.

If a core is not delivered with a physical layout that leverages segmented routing, then its performance will be impacted by interaction with surrounding logic during the place and route phase of the design flow. This yields an unpredictable slow down in the performance of that core.

Figure 3 shows the impact of these affects using a 12x12 multiplier core as a benchmark. With the segmented routing architecture of the XC4000 family and the physical layout provided with the Xilinx LogiCORE multiplier core, performance remains virtually constant even when multiple instances of the core are included on the same device. In

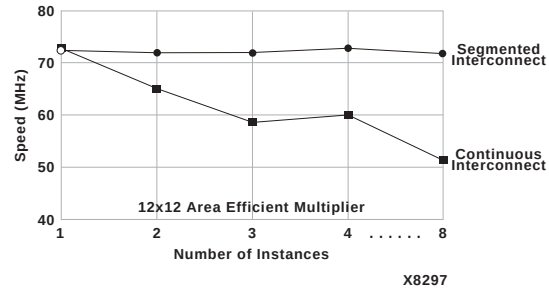

Figure 3: Segmented versus Continuous Interconnect performance for multiple instances of a 12x12 Multiplier Core

contrast, for a PLD with continuous interconnect and cores that only define logic and not layout, note how performance degrades considerably as more cores (or additional user defined logic) are added.

Migrating to a larger device only compounds this problem. Larger devices have longer metal lines with additional capacitance. For a segmented architecture, cores that leverage the interconnect achieve virtually consistent performance regardless of the density of the device. In contrast, cores in a continuous interconnect architecture suffer from a 30 percent performance degradation between the smallest and largest device in the family. This degradation occurs *on top of* that shown in Figure 3.

Performance figures for cores in continuous interconnect architecture PLDs are usually stated for a single instance of that core, implemented in the smallest member of the family in order to show the highest speed. The problem is that it is impossible to know the actual performance of this core until after the design is placed and routed together with all of the other cores and user logic in the target device.

Efficiency

Device utilization is a measure of how efficiently a core uses the logic resources of the PLD. Careful tuning of a design can yield a dramatic reduction in area, resulting in a core that is both very small, and very fast. PLD architecture can have a great impact on this as well.

The availability of abundant routing resources helps a lot. Higher densities can be achieved by offering both global and segmented routing options. Short, isolated signals can be routed onto local interconnect without wasting longer lines. This option is not available with continuous interconnect PLDs.

Many applications, such as in Digital Signal Processing (DSP), take advantage of distributed RAM. PLD-based DSP uses a unique technique known as distributed arithmetic for which a large array of small RAMs is advantageous. PLDs that have no distributed RAM or only block RAM are limited to using flip-flops. In many cases, the sheer number of flip flops required exceeds the number available in the largest device.

For example, a 16-bit shift register that leverages distributed RAM uses only three logic cells in a Xilinx XC4000 device. In a PLD without distributed RAM, the same shift register would take 16 logic cells. This makes functions such as multipliers, transforms and filters take as much as three times the logic resources as they would using distributed RAM. Figure 4 shows comparative examples of this for various 16-bit FIR filters.

Distributed RAM also allows the flexibility to build single- or dual-port memory structures of any size, anywhere in the logic array. Functions like PCI leverage this to obtain 132 Mbyte/sec sustained, zero wait-state burst transfer rates

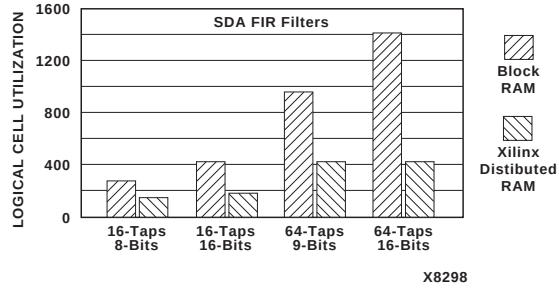

Figure 4: FIR Filter Efficiency in Distributed versus Block RAM PLDs

that are far beyond that which current block-RAM architectures can achieve.

Power Consumption

All device packages have a thermal limit that, when exceeded, will result in thermal breakdown of the device. Since faster clock speeds result in higher power consumption, the thermal limit of the package represents a literal brick wall to the speed at which the chip inside can operate (see Figure 5). Obviously then, a PLD architecture that consumes less power for a given clock frequency can run at a higher speed in the same package.

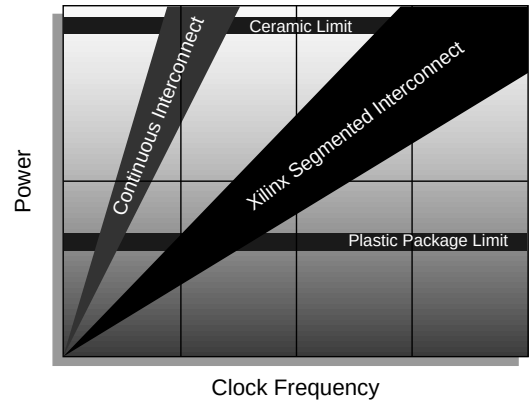

Figure 5: Power Consumption of Segmented versus Continuous Interconnect PLDs

PLDs based on continuous interconnect rely on long metal lines for signal routing. If a signal only needs to travel a short distance on the chip, it must be routed using a longer piece of metal than is necessary, as shown in Figure 2. This introduces additional parasitic capacitance that increases the switching current required. At high frequencies, this causes significantly higher power dissipation. The problem

only gets worse for higher density devices. This is why high density, continuous interconnect PLDs are often only available in large, expensive ceramic packages.

In contrast, a segmented PLD architecture provides the option to route short distance signals on short pieces of metal. This minimizes the load on that signal as well as the current needed to drive it. Cores that are designed using the local interconnect of segmented PLDs will consume less power. This allows the design to run at much higher speeds for a particular package, or to run at the same speed and consume less power than possible using a continuous interconnect architecture. The effect of this is shown in Figure 5. The higher efficiencies achieved through segmented interconnect and distributed RAM further reduce power consumption by allowing a design to be implemented in a smaller device. This allows even higher performance as indicated by the ranges shown in the figure.

Conclusions

The main reason for using cores is to cut development time and allow designers to focus on those portions of the system that they know best. Cores are chosen for features, functionality and performance. When it comes to performance, it is best to know exactly what to expect from a core before the design is started. Anything that increases the possibility of design delays, or even failure, should be avoided.

Xilinx FPGAs, such as the XC4000X architecture, offer capacities reaching the half-million system-gate level. There is no way to maintain the time-to-market value of programmable devices of this capacity efficiently without the use of cores. Xilinx FPGAs offer superior architectural features that allow you to implement cores with guaranteed timing, maximum efficiency and at minimum power allowing higher performance. This comes as the result of silicon features that other PLD architectures cannot alter through upgrades to place and route software.

In addition, all Xilinx LogiCORE functions delivered through the CORE Generator tool are unique in that they include both the logic and layout required to take advantage of the availability of segmented routing, distributed RAM and other architectural features not covered in this paper. This makes them consistent, predictable and fast while minimizing power consumption.

Finally, Xilinx HardWire products provide the fastest available migration to a true high-volume, low-cost ASIC. Conversion to a HardWire device requires no redesign or test vector generation. In this case, the FPGA serves as the prototype and early production vehicle. Once the FPGA design is frozen, Xilinx takes over the conversion process while work can begin on the next design. Only Xilinx owns and offers a risk-free program like this.

No other PLD vendor today offers such a rich set of silicon and software features that makes designing with cores for programmable logic so easy. For more information on Xilinx devices, the CORE Generator tool, or CORE Solutions products, contact your local Xilinx sales representative, send email to logicore@xilinx.com, or look us up on the web at:

http://www.xilinx.com/products/logicore/logicore.htm